International Journal of Neural Systems, Vol. 18, No. 3 (2008) 185–194

World Scientific
www.worldscientific.com

ONLINE CLUSTERING ALGORITHMS

WESAM BARBAKH* and COLIN FYFE†
The University of Paisley, Scotland
**wesam.barbakh@paisley.ac.uk*
†*colin.fyfe@paisley.ac.uk*

We introduce a set of clustering algorithms whose performance function is such that the algorithms overcome one of the weaknesses of K-means, its sensitivity to initial conditions which leads it to converge to a local optimum rather than the global optimum. We derive online learning algorithms and illustrate their convergence to optimal solutions which K-means fails to find. We then extend the algorithm by underpinning it with a latent space which enables a topology preserving mapping to be found. We show visualisation results on some standard data sets.

Keywords: Clustering; K-means.

1. Introduction

One of the major tasks falling to a data analyst is to find groups of data all of which share some underlying feature which is not shared by other samples in the dataset. This is known as clustering the data if it is done in an unsupervised manner i.e. if we do not give the method any class information on which to work. The K-means algorithm[12,15,17] is one of the most frequently used investigatory algorithms in data analysis. The algorithm attempts to locate K prototypes or means throughout a data set in such a way that the K prototypes in some way best represents the data. It is an iterative algorithm in which K means are spread throughout the data and the data samples are allocated to the mean which is closest (often in Euclidean norm) to the sample. Then the K means are repositioned as the average of data points allocated to each mean. This continues until stable convergence is reached. The K-means algorithm is one of the first which a data analyst will use to investigate a new data set because it is algorithmically simple, relatively robust and gives 'good enough' answers over a wide variety of data sets: it will often not be the single best algorithm on any individual data set but it may be close to the optimal over a wide range of data sets. However the algorithm is known to suffer from the defect that the means or prototypes found depend on the initial values given to them at the start of the simulation: a typical program will converge to a local optimum. There are a number of heuristics in the literature which attempt to address this issue but, at heart, the fault lies in the performance function on which K-means is based.

Reference 14 proposed a global K-means algorithm, an incremental approach to clustering that adds one cluster prototype at a time through a deterministic global search consisting of N (the data size) executions of the K-means; this algorithm can obtain equivalent or better results than the standard K-means, but it suffers from high computation cost and at the same time gives no guarantee to find the optimum.

Arthur and Vassilvitskii[3] improved the K-means algorithm by substituting the random allocation of the prototypes with a seeding technique. They give experimental results that show the advantage of this algorithm in time and accuracy.

A variation on K-means is the so-called soft K-means[16] in which prototypes are allocated according to

$$\mathbf{m}_k = \frac{\sum_n r_{kn}\mathbf{x}_n}{\sum_{j,n} r_{jn}} \tag{1}$$

$$\text{where e.g. } r_{kn} = \frac{\exp(-\beta d(\mathbf{x}_n, \mathbf{m}_k))}{\sum_j \exp(-\beta d(\mathbf{x}_n, \mathbf{m}_j))} \tag{2}$$

and $d(a, b)$ is the Euclidean distance between a and b. β is a "stiffness" parameter. Note that the standard K-means algorithm is a special case of the soft K-means algorithm in which the responsibilities, $r_{kn} = 1$ when $\mathbf{m}_k$ is the closest prototype to $\mathbf{x}_n$ and 0 otherwise. However the soft K-means does increase the non-localness of the interaction since the responsibilities are typically never exactly equal to 0 for any data point-prototype combination.

However there are still problems with soft K-means. We find that with soft K-means it is important to choose a good value for β; if we choose a poor value we may have poor results in finding the clusters. However, even with a good value, we often still find that soft K-means has the problem of sensitivity to the prototypes' initialization.

In this paper, we show how we can extend our algorithms in Refs. 5, 7 and allow them to learn in online mode, which may lead to different results due to the different behavior in the learning process. Furthermore, a limitation of batch processing algorithms is that they cannot readily respond to new data if the data only becomes available over time. Also if the data set is non-stationary, a batch algorithm cannot respond to the changing statistics of the data over time. Thus we construct a new set of online clustering algorithms based on extension of some of our algorithms and sharing the same performance functions.

2. A Family of New Algorithms

In this paper we introduce a new family of algorithms that solve the problem of sensitivity to initial conditions in K-means.

Given an input data samples $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, where $\mathbf{x}_i \in \Re^d$, and prototypes $(\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K)$, where $\mathbf{m}_l \in \Re^d$, the performance function for K-means may be written as

$$J_K = \sum_{i=1}^{N} \min_{j=1}^{K} \|\mathbf{x}_i - \mathbf{m}_j\|^2 \tag{3}$$

which we wish to minimise by moving the prototypes to the appropriate positions.

Note that (3) detects only the prototypes closest to data points and then distributes them to give the minimum performance which determines the clustering. Any prototype which is still far from data is not utilised and does not enter any calculation that gives minimum performance. This may result in dead prototypes, which are never appropriate for any cluster. Thus initializing prototypes appropriately can have a big effect in K-means.

The central idea in the new algorithms is that each prototype before moving to any new location takes account of all the other prototypes' positions and, in particular, to their relative locations with respect to the data points, and hence it is possible for it to identify the free clusters that are not recognized by the other prototypes.

2.1. *Weighted K-means algorithm (WK)*

We wish to form a performance function with the following properties:

- Minimum performance gives an intuitively 'good' clustering.
- It creates a relationship between all data points and all prototypes.

We have previously[5] investigated the performance function

$$J_1 = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} \|\mathbf{x}_i - \mathbf{m}_j\| \right] \min_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^2. \tag{4}$$

The rationale behind this performance function is that we wish to utilise the minimum distance in our learning algorithm but retain the global interaction which is necessary to ensure all prototypes play a part in the clustering. We have previously derived a batch clustering algorithm associated with this performance function by calculating the partial derivatives of (4) with respect to the prototypes, setting this to zero and hence finding the optimal prototypes' positions. We call the resulting algorithm Weighted K-means (though recognising that other weighted versions of K-means have been developed in the literature). This gives a solution of

$$\mathbf{m}_r(t+1) = \frac{\sum_{i \in V_r} \mathbf{x}_i a_{ir} + \sum_{i \in V_j, j \neq r} \mathbf{x}_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}} \tag{5}$$

where V_r contains the indices of data points that are closest to $\mathbf{m}_r$, V_j contains the indices of all the other points and

$$a_{ir} = \|\mathbf{x}_i - \mathbf{m}_r(t)\| + 2 \sum_{j=1}^{K} \|\mathbf{x}_i - \mathbf{m}_j\| \tag{6}$$

$$b_{ir} = \frac{\|\mathbf{x}_i - \mathbf{m}_{k*}\|^2}{\|\mathbf{x}_i - \mathbf{m}_r(t)\|} \tag{7}$$

where again $k^* = \arg\min_k \|\mathbf{x}_i - \mathbf{m}_k\|$. We have given extensive analysis and simulations in Ref. 6 showing that this algorithm will cluster the data with the prototypes which are closest to the data points being positioned in such a way that the clusters can be identified. However there can be some potential prototypes which are not sufficiently responsive to the data and so never move to identify a cluster. In fact, these points move to (a weighted) centre of the data set. This may be an advantage in some cases in that we can easily identify redundancy in the prototypes however it does waste computational resources unnecessarily.

2.2. *Inverse Weighted K-means algorithm (IWK)*

We also considered the performance function

$$J_2 = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_j\|^p} \right] \min_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^n. \quad (8)$$

Using the same method as above, this gives the batch algorithm

$$\mathbf{m}_r(t+1) = \frac{\sum_{i \in V_r} \mathbf{x}_i a_{ir} + \sum_{i \in V_j, j \neq r} \mathbf{x}_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}} \quad (9)$$

where V_r contains the indices of data points that are closest to $\mathbf{m}_r$, V_j contains the indices of all the other points and

$$a_{ir} = -(n-p)\|\mathbf{x}_i - \mathbf{m}_r(t)\|^{n-p-2} - n\|\mathbf{x}_i - \mathbf{m}_r(t)\|^{n-2} \sum_{j \neq k*} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_j\|^p} \quad (10)$$

$$b_{ir} = p \frac{\|\mathbf{x}_i - \mathbf{m}_{k*}\|^n}{\|\mathbf{x}_i - \mathbf{m}_r(t)\|^{p+2}}. \quad (11)$$

From the above, we see that $n \geq p$ if the direction of the first term is to be correct and $n \leq p+2$ to ensure stability in all parts of that equation. In practice, we have found that a viable algorithm may be found by using (11) for all prototypes (and thus never using (10) for the closest prototype). We call this the Inverse Weighted K-means Algorithm (IWK).

3. Online Clustering Algorithms

In this section, we show how we can extend the algorithms in Sec. 2 and allow them to learn in online mode. The aim of this section is to allow prototypes to learn in a different way, online, to that in batch mode. This may lead to different results due to the different behavior in the learning process. Furthermore, a limitation of batch processing algorithms is that they cannot readily respond to new data if the data only becomes available over time. Thus we construct a new set of online clustering algorithms based on extension of some of the algorithms in the previous section and sharing the same performance functions.

3.1. *Online K-means algorithm*

The performance function for K-means may be written as

$$J_1 = \sum_{i=1}^{N} \min_{j=1}^{K} \|\mathbf{x}_i - \mathbf{m}_j\|^2 \quad (12)$$

The implementation of the online K-means algorithm is as follows[21]:

(1) Initialization: initialize the cluster prototype vectors $\mathbf{m}_1, \ldots, \mathbf{m}_k$
(2) Loop for M iterations:
   (a) for each data vector $\mathbf{x}_i$, set
   $$k^* = \arg\min_{k=1}^{K}(\|\mathbf{x}_i - \mathbf{m}_k\|)$$
   (b) update the prototype $\mathbf{m}_{k^*}$ as
   $$\mathbf{m}_{k^*}^{(new)} = \mathbf{m}_{k^*} - \zeta \frac{\partial J_1}{\partial \mathbf{m}_{k^*}} = \mathbf{m}_{k^*} + \zeta(\mathbf{x}_i - \mathbf{m}_{k^*})$$
   where ζ is a learning rate usually set to be a small positive number (e.g., 0.05).

The learning rate can also gradually decrease during the learning process.

3.2. *IWK online algorithm v1 (IWKO1)*

As shown in (9), in batch mode we have:

$$\mathbf{m}_k = \frac{a_1\mathbf{x}_1 + \cdots + a_N\mathbf{x}_N}{a_1 + \cdots + a_N} \quad (13)$$

where

$$a_i = \begin{cases} \left(-(n-p)\|\mathbf{x}_i - \mathbf{m}_k\|^{n-p-2} - n\|\mathbf{x}_i - \mathbf{m}_k\|^{n-2} \sum_{j \neq k*} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_j\|^p} \right) & \text{if } \mathbf{m}_k \text{ is closest to } \mathbf{x}_i \\ \left(p \frac{\|\mathbf{x}_i - \mathbf{m}_{k^*}\|^n}{\|\mathbf{x}_i - \mathbf{m}_k\|^{p+2}} \right) & \text{otherwise.} \end{cases}$$

$\mathbf{m}_{k^*}$ is the closest prototype to $\mathbf{x}_i$.

3.2.1. *Implementation (Online mode)*

In online mode, for IWK we can do something similar to (13) (taking into account that we receive one input sample at a time) as follows:

(1) Initialization:
   — initialize the cluster prototype vectors $\mathbf{m}_1, \ldots, \mathbf{m}_K$
   — initialize one dimensional vector, v, with K elements to one, $v_1 = 1, \ldots, v_K = 1$

   note: v_k will represent the value that should be in denominator of (13) after feeding the input sample (it is used also for normalization).

(2) Loop for M iterations:
   — for each data vector $\mathbf{x}_i$, set

$$k^* = \arg\min_{k=1}^{K}(\|\mathbf{x}_i - \mathbf{m}_k\|)$$

   and update all the prototypes $\mathbf{m}_k$ as

$$\mathbf{m}_k^{(new)} = \frac{\mathbf{m}_k v_k + a_i \mathbf{x}_i}{v_k + a_i}$$
$$v_k{}^{(new)} = v_k + a_i$$

To clarify how this implementation is going to build (13), we will feed two data points as an example. After feeding the first data point $\mathbf{x}_1$, we will have the following:

$$\mathbf{m}_k^{(new)} = \frac{(\mathbf{m}_k * 1) + a_1 \mathbf{x}_1}{1 + a_1}$$
$$v_k{}^{(new)} = 1 + a_1$$

After feeding the second data point $\mathbf{x}_2$, we will have the following:

$$\mathbf{m}_k^{(new)} = \frac{\left(\frac{(\mathbf{m}_k * 1) + a_1 \mathbf{x}_1}{1 + a_1}\right)(1 + a_1) + a_2 \mathbf{x}_2}{1 + a_1 + a_2}$$
$$= \frac{\mathbf{m}_k + a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2}{1 + a_1 + a_2}$$
$$v_k{}^{(new)} = 1 + a_1 + a_2$$

And so on in the same way for all data points. In this version we try to implement the batch mode algorithm but when we have the data one at a time. May we need before the end of the execution to subtract $\mathbf{m_k}$ from the numerator and 1 from the denomerator to have (13).

3.2.2. *Simulation*

Figure 1, right, shows the results after applying IWKO1 to the artificial data set in Fig. 1, left.

3.3. *IWK online algorithm v2 (IWKO2)*

In this section we show how it is possible to allow all the units (prototypes) to learn, not only the winner as in K-means or the winner with its neighbors as in SOM. In this algorithm, it is not necessary to specify any functions for the neighbors as all units learn with every input sample.

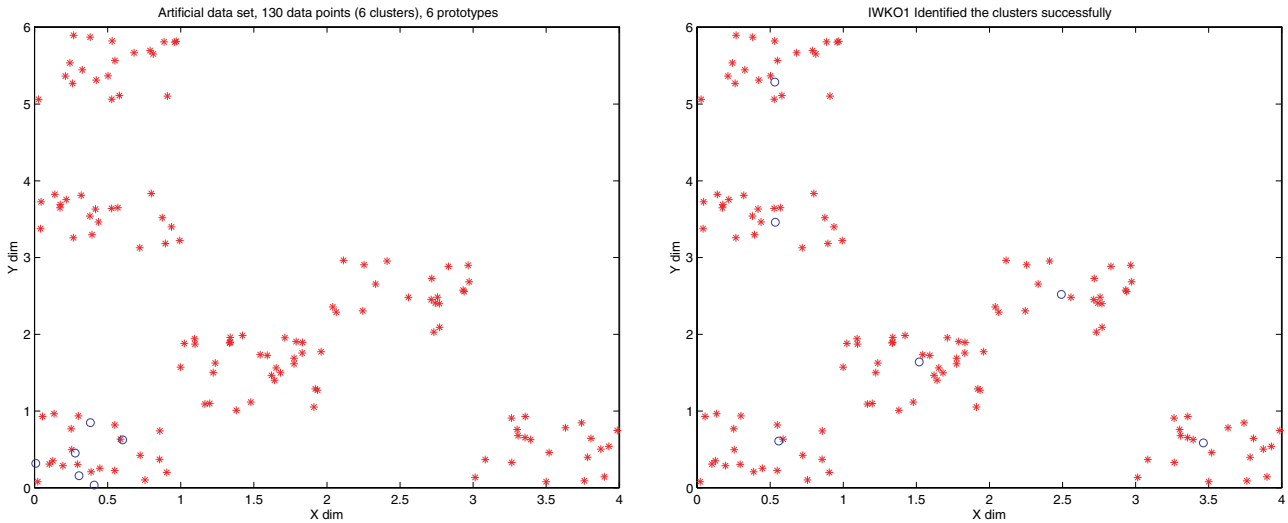

Fig. 1. Left: Artificial data set: data set is shown as 6 clusters of red '*'s, prototypes are initialized to lie within one cluster and shown as blue 'o's. Right: IWKO1 succeeds in identifying all the clusters.

3.3.1. *Implementation (Online mode)*

(1) Initialization: initialize the cluster prototype vectors $\mathbf{m}_1, \ldots, \mathbf{m}_k$
(2) Loop for M iterations:

— for each data vector $\mathbf{x}_i$, set

$$k^* = \arg\min_{k=1}^{K}(\|\mathbf{x}_i - \mathbf{m}_k\|)$$

and update all the prototypes $\mathbf{m}_k$ as

$$\mathbf{m}_{k^*}^{(new)} = \mathbf{m}_{k^*} - \zeta a_{ik^*}(\mathbf{x}_i - \mathbf{m}_{k^*})$$

where

$$a_{ik^*} = -(n+1)\|\mathbf{x}_i - \mathbf{m}_{k^*}\|^{n-1} - n\|\mathbf{x}_i - \mathbf{m}_{k^*}\|^{n-2}\sum_{j\neq k^*}\|\mathbf{x}_i - \mathbf{m}_j\|$$

$$\mathbf{m}_k^{(new)} = \mathbf{m}_k - \zeta\left(\frac{-\|\mathbf{x}_i - \mathbf{m}_{k^*}\|^n}{\|\mathbf{x}_i - \mathbf{m}_k\|}\right) \times (\mathbf{x}_i - \mathbf{m}_k)$$

where ζ is a learning rate usually set to be a small positive number (e.g., 0.05).

3.3.2. *Simulation*

We applied IWKO2 and K-means to some artificial data sets. We found K-means failed to identify all the clusters due to the bad initialization while IWKO2 algorithm succeeded. Figure 3 shows the results after applying K-means, left, and IWKO2, right, to the artificial data set shown in Fig. 2.

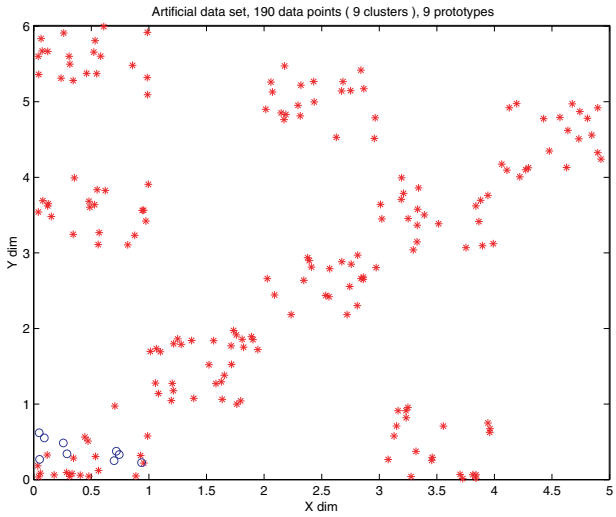

Fig. 2. Artificial data set: data set is shown as 6 clusters of red '*'s, prototypes are initialized to lie within one cluster and shown as blue 'o's.

Note: K-means can succeed with this artificial data set by choosing perfect prototypes' initialization, but we want to show that the new algorithm IWKO2 still works well even if we have a bad initialization in which K-means failed.

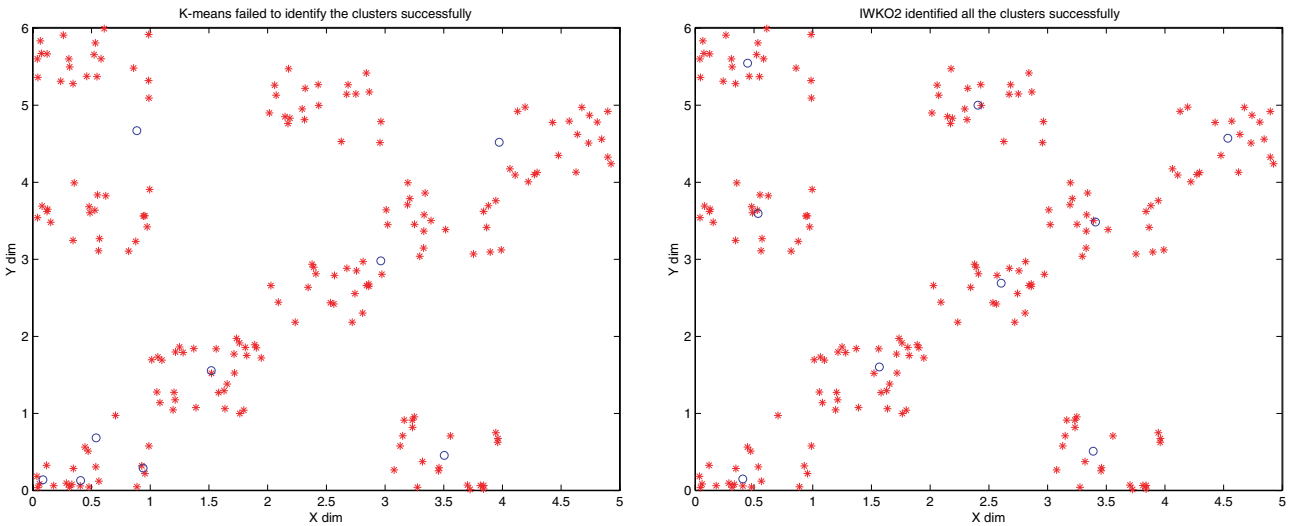

Fig. 3. Left: K-means results. Right: IWKO2 succeeds in identifying all the clusters.

Note: for this algorithm, it is also possible to work in the same manner as the SOM and update the winner with its neighbors instead of updating all the units. In this way, we can create an algorithm that has the advantages of both SOM and IWKO2.

3.4. *K-Harmonic means — online mode algorithm (KHMO)*

In K-Harmonic means we have the following performance function:

$$J_{HA} = \sum_{i=1}^{N} \frac{K}{\sum_{k=1}^{K} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^2}}. \tag{14}$$

Then we wish to move the prototypes in such a way to minimize the performance function and hence identify the clusters

$$\frac{\partial J_{HA}}{\partial \mathbf{m}_k} = -K \sum_{i=1}^{N} \frac{2(\mathbf{x}_i - \mathbf{m}_k)}{\|\mathbf{x}_i - \mathbf{m}_k\|^4 \left\{ \sum_{l=1}^{K} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_l\|^2} \right\}^2} \tag{15}$$

Setting this equal to 0 and "solving" for the $\mathbf{m}_k$'s, we get a recursive formula (Batch Mode)

$$\mathbf{m}_k^{(new)} = \frac{\sum_{i=1}^{N} \frac{1}{d_{i,k}^4 \left(\sum_{l=1}^{K} \frac{1}{d_{i,l}^2} \right)^2} \mathbf{x}_i}{\sum_{i=1}^{N} \frac{1}{d_{i,k}^4 \left(\sum_{l=1}^{K} \frac{1}{d_{i,l}^2} \right)^2}} \tag{16}$$

where we have used $d_{i,k}$ for $\|\mathbf{x}_i - \mathbf{m}_k\|$ to simplify the notation. There are some practical issues to deal with in the implementation details of which are given in Refs. 19, 20.

Reference 20 have extensive simulations showing that this algorithm converges to a better solution (less prone to finding a local minimum because of poor initialization) than both standard K-means or a mixture of experts trained using the EM algorithm.

3.4.1. *Implementation (Online mode)*

From (15) we have:

$$\frac{\partial J_{HA}(\mathbf{x}_i)}{\partial \mathbf{m}_k} = -K \frac{2(\mathbf{x}_i - \mathbf{m}_k)}{\|\mathbf{x}_i - \mathbf{m}_k\|^4 \left\{ \sum_{l=1}^{K} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_l\|^2} \right\}^2} \tag{17}$$

$$\mathbf{m}_k^{(new)} = \mathbf{m}_k + \zeta \left(\frac{2K}{\|\mathbf{x}_i - \mathbf{m}_k\|^4 \{ \sum_{l=1}^{K} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_l\|^2} \}^2} \right) \times (\mathbf{x}_i - \mathbf{m}_k)$$

where ζ is a learning rate usually set to be a small positive number (e.g., 0.05).

3.4.2. *Simulation*

Figure 4 (right), shows the results after applying KHMO to the artificial data set in Fig. 4 (left).

4. A Topology Preserving Mapping

In this section we show how it is possible to extend one of the previous clustering algorithms to provide a new algorithm for visualization and topology-preserving mappings.

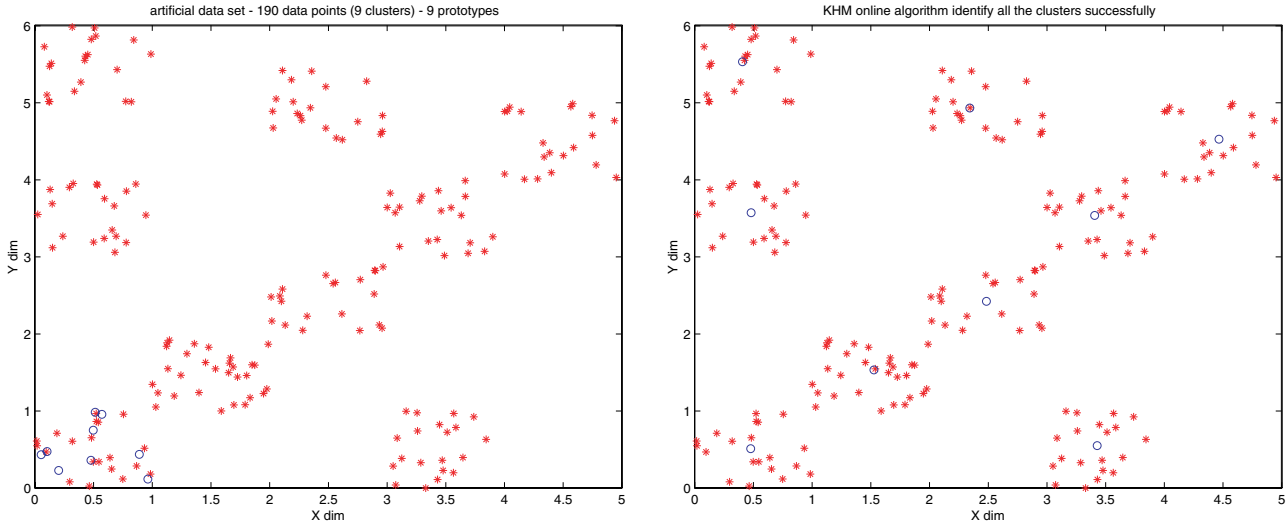

Fig. 4. Left: Artificial data set: data set is shown as 9 clusters of red '*'s, prototypes are initialized to lie within one cluster and shown as blue 'o's. Right: KHMO succeeds in identifying all the clusters.

4.1. *Inverse-weighted K-means (online) Topology-preserving Mapping (IKoToM)*

A topographic mapping (or topology preserving mapping) is a transformation which captures some structure in the data so that points which are mapped close to one another share some common feature while points which are mapped far from one another do not share this feature. The Self-organizing Map (SOM) was introduced as a data quantisation method but has found at least as much use as a visualisation tool.

Topology-preserving mappings such as the Self-organizing Map (SOM)[13] and the Generative Topographic Mapping(GTM)[8] have been very popular for data visualization: we project the data onto the map which is usually two dimensional and look for structure in the projected map by eye. We have recently investigated a family of topology preserving mappings[9] which are based on the same underlying structure as the GTM.

The basis of our model is K latent points, $t_1, t_2, \ldots, t_K$, which are going to generate the K prototypes, $\mathbf{m}_k$. To allow local and non-linear modeling, we map those latent points through a set of M basis functions, $f_1(\,), f_2(\,), \ldots, f_M(\,)$. This gives us a matrix Φ where $\phi_{kj} = f_j(t_k)$. Thus each row of Φ is the response of the basis functions to one latent point, or alternatively we may state that each column of Φ is the response of one of the basis functions to the set of latent points. One of the functions, $f_j(\,)$, acts as a bias term and is set to one for every input. Typically the others are gaussians centered in the latent space. The output of these functions are then mapped by a set of weights, W, into data space. W is $M \times D$, where D is the dimensionality of the data space, and is the sole parameter which we change during training. We will use $\mathbf{w}_i$ to represent the ith column of W and Φ_j to represent the row vector of the mapping of the jth latent point. Thus each basis point is mapped to a point in data space, $\mathbf{m}_j = (\Phi_j W)^T$.

We may update W either in batch mode or with online learning: with the Topographic Product of Experts,[9] we used a weighted mean squared error; with the Harmonic Topographic Mapping,[18] we used Harmonic K-Means. We now apply the Inverse Weighted K-means Online algorithm (IWKO) to the same underlying structure to create a new topology preserving algorithm.

Each data point is visualized as residing at the prototype on the map which would win the competition for that data point. However we can do rather better by defining the responsibility that the jth prototype has for the ith data point as

$$r_{ji} = \frac{\exp(-\gamma\|\mathbf{x}_i - \mathbf{w}_j\|^2)}{\sum_k \exp(-\gamma\|\mathbf{x}_i - \mathbf{w}_k\|^2)} \tag{18}$$

We then project points taking into account these responsiblities: let y_{ij} be the projection of the ith data point onto the jth dimension of the latent space; then

$$y_{ij} = \sum_k t_{kj} r_{ki} \tag{19}$$

where t_{kj} is the jth coordinate of the kth latent point.

4.2. *Inverse-weighted K-means (batch mode) Topology-preserving Mapping (IKToM)*

The IKToM algorithm, like teh IKoToM algorithm, has the same underlying structure as the GTM, with a number of latent points that are mapped to a feature space by M Gaussian functions, and then into the data space by a matrix W. Each latent point t indexed by k is mapped, through a set of M fixed basis functions $\phi_1(\,), \phi_2(\,), \ldots, \phi_M(\,)$ to a prototype in data space $m_k = W\phi(t_k)$.

But the similarity ends there because the objective function is not a probabilistic function like the GTM neither it is optimised with the Expectation-Maximization (EM) algorithm. Instead, the IKToM uses the well proved clustering abilities of the K-means algorithm, improved by using Inverse Weighted K-means (IWK), batch mode, to make it insensitive to initialisation.

4.3. *Simulation*

4.3.1. *Artificial data set*

We create a simulation with 10 latent points deemed to be equally spaced in a one dimensional latent space, passed through 5 Gaussian basis functions and then mapped to the data space by the linear mapping W which is the only parameter we adjust. We generated 500 two dimensional data points, (x_1, x_2), from the function $x_2 = x_1 + 1.25\sin(x_1) + \mu$ where μ is noise from a uniform distribution in [0,1]. Final result from the IKoToM is shown in Fig. 5 in which the projections of consecutive latent points are joined. We

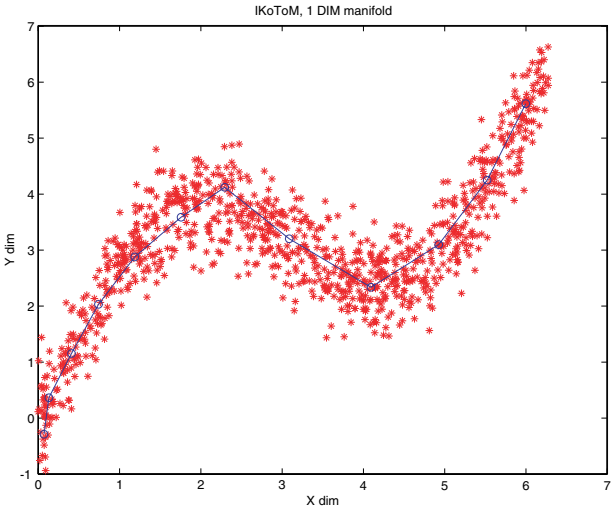

Fig. 5. The resulting prototypes' positions after applying IKoToM. Prototypes are shown as blue 'o's.

see that nearby latent points take responsibility for nearby data points.

4.3.2. *Real data set*

(1) IRIS data set:
In this data set we have 150 samples with 4 dimensions and 3 types.
The iris data set is available at Ref. 1.

(2) Algae data set:
In this data set we have 72 samples with 18 dimensions and 9 types.
This data set is available at Ref. 1.

(3) Genes data set:
In this data set we have 40 samples with 3036 dimensions and 3 types.
This data set is available at Ref. 2.

(4) Glass data set:
In this data set we have 214 samples with 10 dimensions and 6 types.
This data set is available at Ref. 1.

We show in Figs. 6, 7, 8 and 9 the projections of the real data set shown above onto a two dimensional grid of latent points using IKoToM, left, and IKToM, right. In each case, we see a projection of the classes into the latent space which gives a separation between the classes: most researchers will recognise the projection of the iris data set as familiar in that one class is readily separated while the other two are rather more difficult to separate; on the other data sets, we achieve results which are comparable with the best projections found by other methods we have used,[4,10,11] for example, by using IKoToM we identify all the 9 clusters in algae data set, while in the other methods 7 clusters or less were identified.

5. Conclusion

We have discussed one shortcoming of the K-means algorithm: its sensitivity to poor initialization which leads it to converge to a local rather than global optimum. We have shown how different performance functions lead to algorithms which incorporate the correct mixture of local and global knowledge to

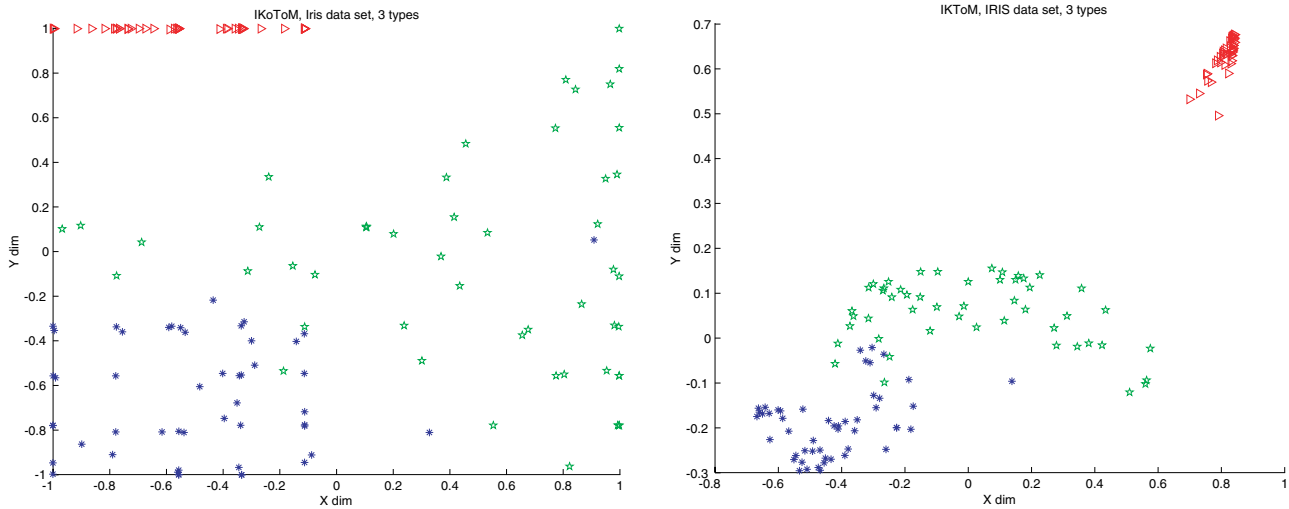

Fig. 6. The result of applying IKoToM, left, and IKToM, right to the Iris real data set.

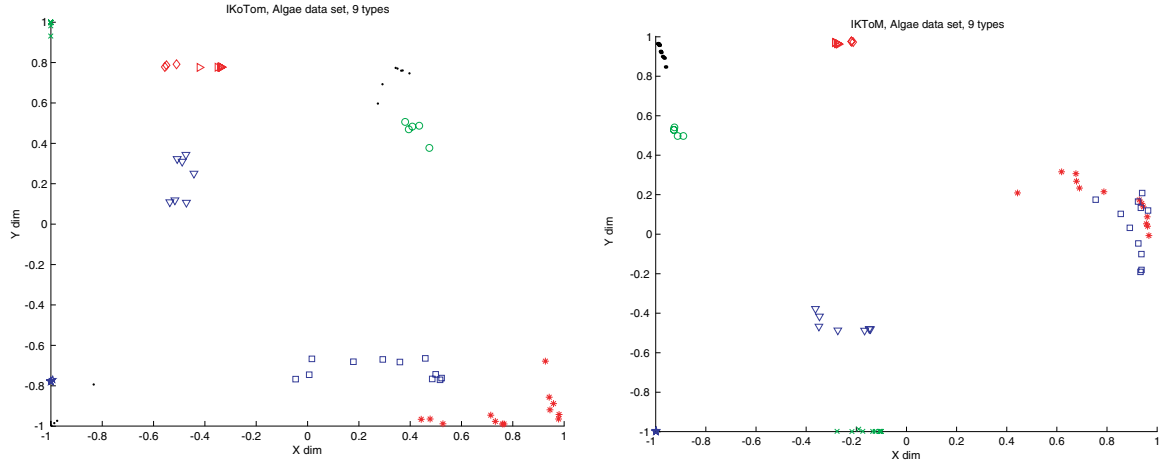

Fig. 7. The result of applying IKoToM, left, and IKToM, right to the Algae real data set.

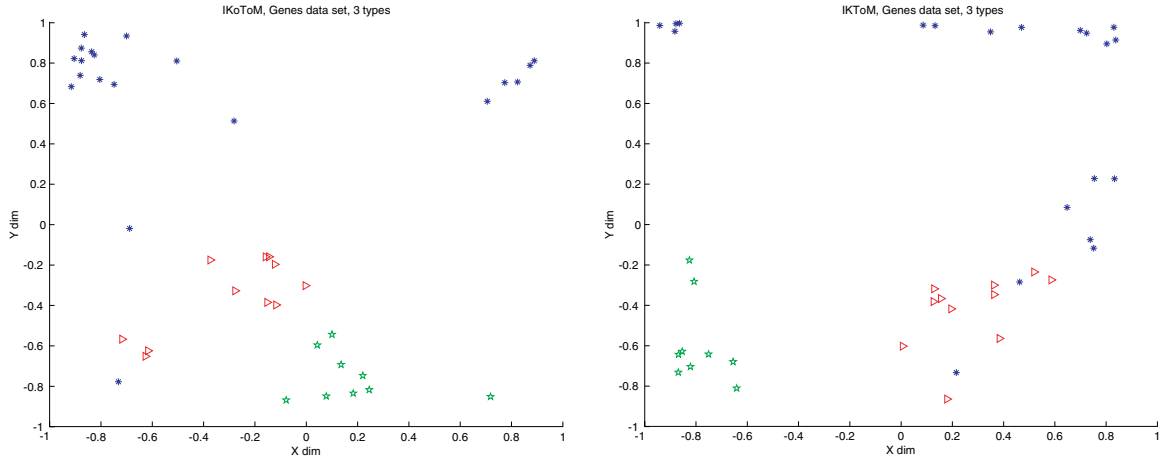

Fig. 8. The result of applying IKoToM, left, and IKToM, right to the Genes real data set.

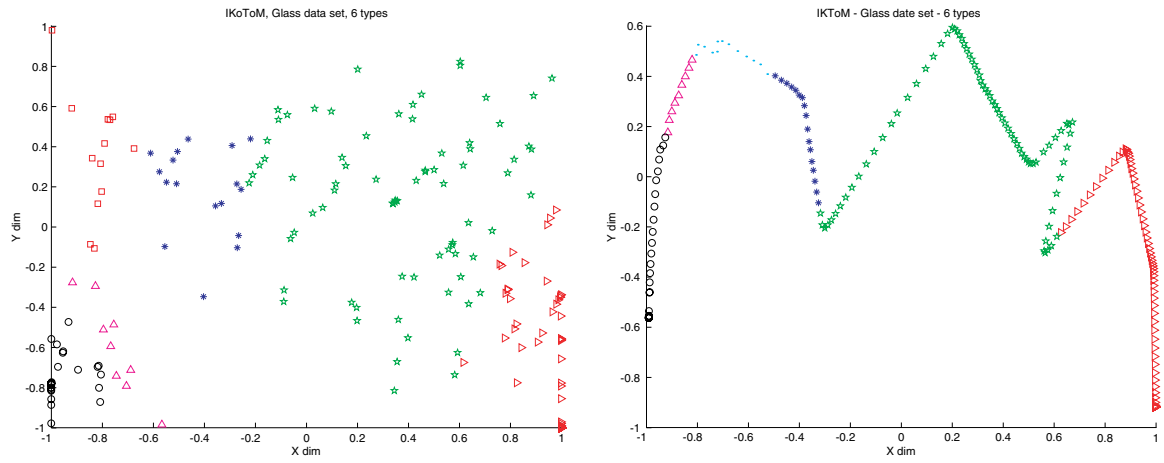

Fig. 9. The result of applying IKoToM, left, and IKToM, right to the Glass real data set.

allow prototypes to optimally cluster a data set. We have derived both a batch algorithm and online algorithms from these performance functions.

We have extended these algorithms by using them with an underlying latent space which enables topology preserving mappings to be developed. We have illustrated these mappings (both batch and online versions) on a variety of data sets and shown how they may be used to visualise these data sets.

References

1. http://mlearn.ics.uci.edu/databases/.
2. http://www.ihes.fr/∼zinovyev/princmanif2006/.
3. D. Arthur and S. Vassilvitskii, K-means++: The advantages of careful seeding. *In Bay Area Theory Symposium, BATS 06*, (2006). http://www.stanford.edu/sergeiv/papers/kMeansPP-soda.pdf.
4. W. Barbakh, The family of inverse exponential k-means algorithms, *Computing and Information Systems* **11**(1) (February 2007) 1–10. (ISSN 1352-9404).
5. W. Barbakh, M. Crowe and C. Fyfe, A family of novel clustering algorithms, in *7th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL2006* (September 2006) 283–290. (ISSN 0302-9743 ISBN-13 978-3-540-45485-4).
6. W. Barbakh and C. Fyfe, Performance functions and clustering algorithms, *Computing and Information Systems* **10**(2) (May 2006) 2–8. (ISSN 1352-9404).
7. W. Barbakh and C. Fyfe, Clustering with reinforcement learning, in *International Conference on Intelligent Data Engineering and Automated Learning IDEAL'07* (December 2007) 507–516. (LNCS 4881).
8. C. M. Bishop, M. Svensen, and C. K. I. Williams, GTM: The generative topographic mapping, *Neural Computation* (1997).
9. C. Fyfe, Two topographic maps for data visualization, *Data Mining and Knowledge Discovery* **14** (2007) 207–224. (ISSN 1384-5810).
10. C. Garcia-Osorio and C. Fyfe, The combined use of self-organising maps and andrews' curves, *International Journal of Neural Systems* (2005).
11. C. Garcia-Osorio and C. Fyfe, Visualisation of high dimensional data via orthogonal curves, *Journal of Universal Computer Science* **11** (2005).
12. J. Hartigan and M. Wang, A k-means clustering algorithm, *Applied Statistics* **28** (1979) 100–108.
13. T. Kohonen, *Self-Organising Maps* (Springer, 1995).
14. A. Likas, N. Vlassis and J. J. Verbeek, The global k-means clustering algorithm, *Pattern Recognition* **36** (2003) 451–461.
15. S. P. Lloyd, Least squares quantization in PCM, Technical note, Bell Laboratories (1957). [Published in 1982 in IEEE Transactions on Information Theory 28, 128–137].
16. D. J. MacKay, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, 2003).
17. J. MacQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967) 281–297.
18. M. Peña and C. Fyfe, Model- and data-driven harmonic topographic maps, *WSEAS Transactions on Computers* **4**(9) (2005) 1033–1044.
19. B. Zhang, Generalized k-harmonic means — boosting in unsupervised learning, Technical Report HPL-2000-137, HP Laboratories, Palo Alto (October 2000).
20. B. Zhang, M. Hsu and U. Dayal, K-harmonic means — a data clustering algorithm. Technical Report HPL-1999-124, HP Laboratories, Palo Alto (October 1999).
21. S. Zhong, T. Khoshgoftaar and N. Seliya, Clustering-based network intrusion detection, *International Journal of Reliability, Quality and Safety Engineering* (2005).